Raphaël **Hertzog**
Roland **Mas**

Cahiers de l'Admin

Debian Lenny

Collection dirigée par Nat **Makarévitch**

ISBN : 978-2-212-12443-9

EYROLLES

Maintenance et mise à jour : les outils APT

Ce qui rend Debian si populaire auprès des administrateurs, c'est la facilité avec laquelle il est possible d'y installer des logiciels et de mettre à jour le système complet. Cet avantage unique est dû en grande partie au programme *APT*, outil dont les administrateurs de Falcot SA se sont empressés d'étudier les possibilités.

SOMMAIRE

- Renseigner le fichier sources.list
- Commande apt-get
- Commande apt-cache
- Frontaux : aptitude, synaptic, gnome-apt
- Vérification d'authenticité des paquets
- Mise à jour d'une distribution à la suivante
- Maintenir un système à jour
- Mise à jour automatique
- Recherche de paquets

MOTS-CLÉS

- apt-get
- apt-cache
- aptitude
- synaptic
- sources.list
- apt-cdrom

VOCABULAIRE
Source de paquets et paquet source

Le terme *source* est source d'ambiguïté. Il ne faut pas confondre un paquet source — paquet contenant le code source d'un programme — et une source de paquets — emplacement (site web, serveur FTP, CD-Rom, répertoire local, etc.) contenant des paquets.

B.A.-BA
Compression gzip et bzip2

Une extension `.gz` dénote un fichier compressé avec l'utilitaire **`gzip`**. De la même manière, `.bz2` indique une compression par **`bzip2`**. **`gzip`** est l'utilitaire Unix traditionnel pour compresser les fichiers, rapide et efficace. **`bzip2`**, plus récent, obtient de meilleurs taux de compression mais nécessite plus de temps de calcul pour comprimer un fichier.

APT est l'abréviation de *Advanced Package Tool* (outil avancé pour les paquets). Ce que ce programme a d'« avancé », c'est la manière d'aborder la problématique des paquets. Il ne se contente pas de les évaluer un par un, mais les considère dans leur ensemble et réalise la meilleure combinaison possible de paquets en fonction de tout ce qui est disponible et compatible (au sens des dépendances).

APT a besoin qu'on lui fournisse une « liste de sources de paquets » : c'est le fichier `/etc/apt/sources.list` qui décrira les différents emplacements (ou « sources ») publiant des paquets Debian. APT devra ensuite rapatrier la liste des paquets publiés par chacune de ces sources, ainsi que leurs en-têtes. Il réalise cette opération en téléchargeant les fichiers `Packages.gz` ou `Packages.bz2` (cas d'une source de paquets binaires) et `Sources.gz` ou `Sources.bz2` (cas d'une source de paquets sources) et en analysant leur contenu. Lorsque l'on dispose déjà d'une copie ancienne de ces fichiers, APT est capable de les mettre à jour en ne téléchargeant que les différences (voir encadré « Mise à jour incrémentale » page 95).

Renseigner le fichier sources.list

Le fichier `/etc/apt/sources.list` contient sur chaque ligne active une description de source, qui se décompose en 3 parties séparées par des blancs.

Le premier champ indique le type de la source :

- « `deb` » pour des paquets binaires,
- « `deb-src` » pour des paquets sources.

Le deuxième champ indique l'URL de base de la source (combinée aux noms de fichier présents dans les fichiers `Packages.gz`, elle doit donner une URL complète valide) : il peut s'agir d'un miroir Debian ou de toute autre archive de paquets mise en place par des tierces personnes. L'URL peut débuter par `file://` pour indiquer une source locale située dans l'arborescence de fichiers du système, par `http://` pour indiquer une source accessible depuis un serveur web, ou encore par `ftp://` pour une source disponible sur un serveur FTP. On trouvera aussi `cdrom://` pour les installations à partir de CD-Rom, mais moins fréquemment, les méthodes d'installation par le réseau étant de plus en plus répandues.

Le dernier champ a une syntaxe variable selon que la source correspond à un miroir Debian ou non. Dans le cas d'un miroir Debian, on nomme la distribution choisie (`stable`, `testing`, `unstable` ou leurs noms de code du moment — voir la liste dans l'encadré « COMMUNAUTÉ » page 8) puis les différentes sections souhaitées (choisies parmi `main` , `contrib`, et `non-free`). Dans les autres cas, on indique simplement le sous-répertoire de la

source désirée (on y trouve souvent le simple « `./` » dénotant l'absence de sous-répertoire — les paquets sont alors directement à l'URL indiquée).

D'une manière générale, le contenu d'un fichier `sources.list` standard pourrait être le suivant :

EXEMPLE Fichier /etc/apt/sources.list

```
# Mises à jour de sécurité
deb http://security.debian.org/ stable/updates main contrib non-free
deb-src http://security.debian.org/ stable/updates main contrib non-free

# Miroir Debian
deb http://ftp.fr.debian.org/debian stable main contrib non-free
deb-src http://ftp.fr.debian.org/debian stable main contrib non-free
```

VOCABULAIRE Les archives main, contrib et non-free

Debian prévoit trois sections pour différencier les paquets selon les licences prévues par les auteurs des programmes respectifs. *Main* (archive principale) rassemble tous les paquets répondant pleinement aux principes du logiciel libre selon Debian.
L'archive *non-free* (non libre), spéciale, contient des logiciels ne répondant pas (totalement) à ces principes mais néanmoins distribuables librement. Cette archive, qui ne fait pas officiellement partie de Debian, est un service rendu aux utilisateurs qui pourraient avoir besoin de ses logiciels — mais Debian recommande toujours d'accorder la préférence aux logiciels libres. L'existence de cette section gêne considérablement Richard M. Stallman et empêche la Free Software Foundation de recommander l'usage de Debian.
Contrib (contributions) est un stock de logiciels libres ne fonctionnant pas sans certains éléments non libres. Il peut s'agir de programmes dépendant de logiciels de la section *non-free* ou de fichiers non libres tels que des ROM de jeux, des BIOS de consoles, etc. On y trouve encore des logiciels libres dont la compilation nécessite des éléments propriétaires. C'était au début le cas de la suite bureautique OpenOffice.org, qui avait besoin d'un environnement Java propriétaire.

Ce fichier référence toutes les sources de paquets associées à la version stable de Debian. Si vous souhaitez utiliser *Testing* ou *Unstable*, il faudra évidemment y ajouter (ou les remplacer par) les lignes adéquates. Lorsque la version désirée d'un paquet est disponible sur plusieurs miroirs, c'est celui qui est listé en premier dans le fichier `sources.list` qui sera employé. Pour cette raison on préfère ajouter les sources non-officielles à la fin du fichier.

Le fichier `sources.list` comporte encore d'autres types d'entrées : celles décrivant des CD-Rom Debian dont vous disposez. Contrairement aux autres entrées, un CD-Rom n'est pas disponible en permanence puisqu'il faut l'insérer dans le lecteur et qu'un seul disque peut être lu à la fois — ces sources sont donc gérées un peu différemment. On ajoutera ces entrées à l'aide du petit programme **`apt-cdrom`**, habituellement invoqué avec le paramètre `add`. Ce dernier demande alors d'insérer le disque dans le lecteur et parcourt son contenu à la recherche de fichiers `Packages`,

DÉCOUVERTE apt-spy

Ce logiciel teste la vitesse de téléchargement depuis plusieurs miroirs Debian et génère un fichier `sources.list` pointant sur le miroir le plus rapide.
Le miroir sélectionné pendant l'installation convient généralement puisqu'il est choisi en fonction du pays. Mais si jamais l'on constatait des lenteurs lors du téléchargement, il sera peut-être utile d'essayer cette application disponible dans le paquet `apt-spy`.

CULTURE La section non-US

Historiquement (jusqu'à *Woody*), il existait un jeu d'archives complémentaires, dites archives *non-US*, également subdivisées en *main*, *contrib* et *non-free*. Ces archives, utilisées en complément des principales, avaient pour vocation de contenir certains paquets soumis à des réglementations spécifiques. Il s'agissait principalement de paquets contenant des programmes de cryptographie, dont l'export depuis les États-Unis devait passer par une autorisation officielle. Ces archives étaient donc hébergées en dehors des États-Unis, de sorte que le téléchargement de ces paquets ne constituait pas un export. Les lois américaines ayant été assouplies de ce côté-là, *non-US* a disparu dans *Etch*, après avoir été entièrement vide pour *Sarge*.
N'oubliez donc pas de vérifier votre `sources.list` pour en supprimer les lignes dorénavant inutiles !

POUR ALLER PLUS LOIN Les paquets d'Experimental

L'archive de paquets *Experimental*, présente sur tous les miroirs Debian, contient des paquets qui n'ont pas encore leur place dans la version *Unstable* pour cause de qualité insuffisante — ce sont fréquemment des versions de développement ou pré-versions (alpha, bêta, *release candidate*...) des logiciels. Il arrive également qu'un paquet y soit envoyé après avoir subi des changements importants, potentiellement sources de problèmes. Le mainteneur cherche alors à débusquer ceux-ci avec l'aide des utilisateurs avancés capables de gérer les soucis importants. Après cette première phase, le paquet passe dans *Unstable*, au public beaucoup plus vaste, et où il subira donc des tests de bien plus grande envergure.
On réservera donc *Experimental* aux utilisateurs qui n'ont pas peur de casser leur système puis de le réparer. Cette distribution peut quand même permettre de rapatrier ponctuellement un paquet que l'on tient à essayer ou utiliser. C'est d'ailleurs la logique standard que Debian lui associe, puisque son ajout dans le fichier `sources.list` d'APT n'entraîne pas l'emploi systématique des paquets qui s'y trouvent. La ligne qu'il convient d'ajouter est la suivante :

```
deb http://ftp.fr.debian.org/debian experimental main contrib non-free
```

qu'il utilisera pour mettre à jour sa base de données de paquets disponibles (opération habituellement réalisée par la commande **`aptitude update`**). Dès lors, **`APT`** pourra vous demander d'insérer le CD-Rom en question s'il a besoin de l'un de ses paquets.

Ressources non officielles : apt-get.org, mentors.debian.net et backports.org

Il existe de nombreuses sources non officielles de paquets Debian, mises en place par des utilisateurs avancés ayant recompilé certains logiciels, par des programmeurs mettant leur création à disposition, et même par des développeurs Debian proposant des pré-versions de leur paquet en ligne. Un site web fut mis en place pour trouver plus facilement ces sources alternatives. On y trouve une quantité impressionnante de sources de paquets Debian prêtes à être intégrées dans les fichiers `sources.list`. Attention toutefois à ne pas rajouter n'importe quoi. Chaque source est en effet prévue pour une version particulière de Debian (celle employée pour compiler les paquets concernés) ; on veillera à maintenir une certaine cohérence dans ce que l'on choisit d'installer.

▸ http://www.apt-get.org/

Signalons également l'existence du site `mentors.debian.net`, qui regroupe des paquets réalisés par des prétendants au statut de développeur Debian officiel ou par des volontaires souhaitant créer des paquets Debian sans passer par ce processus d'intégration. Ces paquets sont donc fournis sans aucune garantie de qualité ; prenez garde à vous assurer de leur origine et intégrité puis à bien les tester avant d'envisager de les déployer.

COMMUNAUTÉ Les sites en debian.net

Le domaine *debian.net* ne constitue pas une ressource officielle du projet Debian. Chaque développeur Debian a la possibilité d'employer ce nom de domaine pour l'usage de son choix. On y trouve des services officieux (parfois des sites personnels) hébergés sur une machine n'appartenant pas au projet et mis en place par des développeurs Debian, voire des prototypes attendant d'être migrés sur *debian.org*. Deux raisons peuvent expliquer que certains de ces prototypes restent en *debian.net* : soit personne ne souhaite faire l'effort nécessaire à sa transformation en service officiel (hébergé dans le domaine *debian.org*, et avec une certaine garantie de maintenance), soit le service est trop controversé pour être officialisé.

Installer un paquet revient à donner les droits root à son concepteur, car il décide du contenu des scripts d'initialisation qui sont exécutés sous cette identité. Les paquets officiels Debian sont réalisés par des volon-

taires cooptés et examinés, capables de sceller leurs paquets pour en vérifier l'origine et l'intégrité.

Mais défiez-vous a priori d'un paquet dont l'origine est incertaine et qui n'est pas hébergé sur un des serveurs officiels du projet Debian : évaluez le degré de confiance que vous accordez au concepteur et vérifiez l'intégrité du paquet.

▸ http://mentors.debian.net/

Enfin, il existe également un site regroupant des « rétroportages » (*backports*) de paquets. Ce terme désigne un paquet d'un logiciel récent recompilé pour une distribution plus ancienne, généralement *Stable*. Lorsque cette distribution commence à dater, de nombreux logiciels évoluent en amont, et les nouvelles versions ne sont pas réintégrées dans la *Stable* courante (qui n'est modifiée que pour prendre en compte les problèmes les plus critiques, comme les problèmes de sécurité). Comme les distributions *Testing* et *Unstable* peuvent être plus risquées, des volontaires proposent parfois des recompilations des logiciels récents pour *Stable*, ce qui permet de restreindre une éventuelle instabilité à un petit nombre, bien choisi, de paquets. Ces paquets sont souvent publiés sur un même site ; là encore, on évaluera le degré de confiance et le risque avant d'installer un paquet en provenance de ce site.

▸ http://www.backports.org/

Commandes aptitude et apt-get

APT est un projet relativement vaste, qui prévoyait à l'origine une interface graphique. Il repose sur une bibliothèque contenant le cœur de l'application, et **`apt-get`** est la première interface — en ligne de commande — développée dans le cadre du projet.

De nombreuses interfaces graphiques sont ensuite apparues en tant que projets extérieurs : **`synaptic`**, **`gnome-apt`**, **`aptitude`** (mode texte), **`wajig`**, etc. Le frontal le plus recommandé, **`aptitude`**, est celui employé lors de l'installation de Debian. Sa syntaxe en ligne de commande, très similaire à celle d'**`apt-get`**, explique notre choix de l'employer dans les exemples de cette section. En cas de différences notables avec **`apt-get`**, celles-ci seront détaillées.

Initialisation

Un préalable à tout travail avec APT est la mise à jour de la liste des paquets disponibles, qui s'effectue avec un simple **`aptitude update`**. Selon le débit de votre connexion, cette opération peut durer puisqu'elle télécharge un certain nombre de fichiers `Packages.(gz|bz2)` (voire `Sources.(gz|bz2)`), devenus assez volumineux au fil de la croissance de Debian (plus de 5 Mo

pour le plus gros `Packages.gz`, correspondant à la section `main`). Évidemment, une installation à partir d'un jeu de CD-Rom ne nécessite aucun téléchargement — cette opération est alors très rapide.

Installation et suppression

APT permet d'ajouter ou de supprimer des paquets sur le système, respectivement avec **`aptitude install`** ***`paquet`*** et **`aptitude remove`** ***`paquet`***. Dans chaque cas, APT installera automatiquement les dépendances nécessaires ou supprimera les paquets dépendant du paquet en cours de désinstallation. La commande **`aptitude purge`** ***`paquet`*** demande une désinstallation complète — les fichiers de configuration sont alors également supprimés. Purger un paquet avec **`apt-get`** se réalise avec la commande **`remove`** à laquelle on associe l'option **`--purge`**.

ASTUCES

Installer la même sélection de paquets plusieurs fois

Il est parfois souhaitable de pouvoir installer systématiquement la même liste de paquets sur plusieurs ordinateurs. C'est possible assez facilement.
Récupérons d'abord la liste des paquets installés sur l'ordinateur qui servira de « modèle » à dupliquer.

```
$ dpkg --get-selections >liste-pkg
```

Le fichier `liste-pkg` contient alors la liste des paquets installés.
Il faut alors transférer le fichier `liste-pkg` sur les ordinateurs à mettre à jour et y employer les commandes suivantes :

```
# dpkg --set-selections <liste-pkg
# apt-get dselect-upgrade
```

La première commande enregistre les vœux de paquets à installer, que l'invocation d'**`apt-get`** exauce ensuite ! **`aptitude`** n'offre pas cette commande.

Supprimer et installer en même temps

Il est possible, en ajoutant un suffixe, de demander à **`aptitude`** d'installer certains paquets et d'en supprimer d'autres sur la même ligne de commande. Lors d'une commande **`aptitude install`**, ajoutez un « - » aux noms des paquets que vous souhaitez supprimer. Lors d'une commande **`aptitude remove`**, ajoutez un « + » aux noms des paquets que vous souhaitez installer.

L'exemple suivant montre deux manières d'installer *paquet1* et de supprimer *paquet2*.

```
# aptitude install paquet1 paquet2-
[...]
# aptitude remove paquet1+ paquet2
[...]
```

apt-get --reinstall et aptitude reinstall

Il arrive que le système soit endommagé suite à la suppression ou à la modification de fichiers appartenant à un paquet. Le moyen le plus simple de récupérer ces fichiers est alors de réinstaller le paquet concerné. Malheureusement, le système de paquetage considère que ce dernier est déjà installé et refuse poliment de s'exécuter ; l'option `--reinstall` de la commande **`apt-get`** permet précisément d'éviter cet écueil. La commande ci-dessous réinstalle `postfix` même si ce dernier est déjà présent.

```
# apt-get --reinstall install postfix
```

La ligne de commande d'**`aptitude`** est un peu différente, mais le même effet s'obtient avec **`aptitude reinstall postfix`**.
Le problème ne se pose pas avec **`dpkg`**, mais il est rare que l'administrateur emploie directement ce dernier.
Attention, recourir à **`apt-get --reinstall`** pour restaurer des paquets modifiés au cours d'une attaque ne suffit certainement pas à retrouver un système identique à ce qu'il était au préalable. Le chapitre 14 détaille la marche à suivre en cas de piratage (page 360).

Si le fichier `sources.list` mentionne plusieurs distributions, il est possible de préciser la version du paquet à installer. On peut demander un numéro de version précis avec **`aptitude install`** ***`paquet=version`***, mais on se contentera en général d'indiquer la distribution d'origine du paquet (*Stable*, *Testing* ou *Unstable*) avec la syntaxe **`aptitude install`** ***`paquet/distribution`***. Avec cette commande, on pourra donc revenir à une ancienne version d'un paquet (si par exemple on sait qu'elle fonctionne bien), à condition qu'elle soit encore disponible dans une des sources référencées par le fichier `sources.list`.

EXEMPLE **Installation de la version Unstable de spamassassin**

```
# aptitude install spamassassin/unstable
```

Mise à jour

Des mises à jour régulières sont recommandées, car elles mettront en place les derniers correctifs de sécurité. Pour cela, on invoquera **`aptitude safe-upgrade`** ou **`apt-get upgrade`** (évidemment précédé par **`aptitude update`**). Cette commande cherche les mises à jour des paquets installés, réalisables sans supprimer de paquets. Autrement dit, l'objectif est d'assurer une mise à jour la moins intrusive possible. Pour cette action, **`apt-get`** est un peu plus exigeant que **`aptitude`** parce qu'il refusera d'installer des paquets qui n'étaient pas préalablement installés.

ASTUCE **Mise à jour incrémentale**

On l'a vu, le but de la commande **`aptitude update`** est de télécharger pour chacune des sources de paquets le fichier `Packages` (ou `Sources`) correspondant. Cependant, même après compression **`bzip2`**, ces fichiers restent volumineux (le `Packages.bz2` pour la section *main* de *Lenny* occupe plus de 5 Mo). Si l'on souhaite effectuer des mises à jour régulières, ces téléchargements peuvent prendre du temps inutilement.

Une nouveauté depuis *Etch* est qu'APT peut dorénavant télécharger non plus le fichier entier mais simplement les différences par rapport à une version précédente. Les miroirs Debian officiels distribuent pour cela différents fichiers recensant les différences d'une version du fichier `Packages` à la suivante, lors des mises à jour des archives, avec un historique d'une semaine. Chacun de ces fichiers de différences ne pesant en général que quelques dizaines de kilo-octets pour *Unstable*, la quantité de données téléchargées par un **`aptitude update`** hebdomadaire est typiquement divisée par 10. Pour les distributions moins mobiles, comme *Stable* et *Testing*, le gain est encore plus flagrant.

On notera cependant qu'il est parfois intéressant de forcer le téléchargement du fichier `Packages.bz2` complet, notamment lorsque la dernière mise à jour est vraiment trop ancienne et que le mécanisme des différences incrémentales n'apporterait rien. Cela peut également être intéressant dans les cas où l'accès réseau est très rapide mais le processeur de la machine à mettre à jour est relativement lent, le temps gagné sur le téléchargement des fichiers étant plus que perdu lors du calcul des nouvelles versions de ces fichiers à partir des anciennes versions et des différences téléchargées. Pour cela, on pourra utiliser le paramètre de configuration `Acquire::Pdiffs`, que l'on règlera à `false`.

POUR ALLER PLUS LOIN
Cache des fichiers .deb

APT conserve dans le répertoire `/var/cache/apt/archives/` une copie de chaque fichier `.deb` téléchargé. Dans le cas de mises à jour fréquentes, ce répertoire peut rapidement occuper beaucoup d'espace disque avec plusieurs versions de chaque paquet ; il convient donc d'y faire régulièrement le tri. Deux commandes existent pour cela : **`aptitude clean`** vide le répertoire ; **`aptitude autoclean`** ne supprime que les paquets qui, n'étant plus téléchargeables (car ayant disparu du miroir Debian), sont clairement inutiles (le paramètre de configuration `APT::Clean-Installed` permet d'empêcher la suppression de fichiers `.deb` encore actuellement installés).

Remarquons cependant qu'**aptitude** retiendra en général le numéro de version le plus récent (à l'exception des paquets *Experimental*, ignorés par défaut quel que soit leur numéro de version). Si vous avez mentionné *Testing* ou *Unstable* dans votre `sources.list`, **aptitude safe-upgrade** migrera une grande partie de votre système *Stable* en *Testing* ou *Unstable*, ce qui n'est peut-être pas l'effet recherché.

Pour indiquer à **aptitude** d'utiliser telle ou telle distribution pour ses recherches de paquets mis à jour, il faut utiliser l'option `-t` ou `--target-release` (version cible), suivie du nom de la distribution en question (exemple : **aptitude -t stable safe-upgrade**). Pour éviter de spécifier cette option à chaque invocation d'**aptitude**, vous pouvez ajouter `APT::Default-Release "stable";` dans le fichier `/etc/apt/apt.conf.d/local`.

Pour les mises à jour plus importantes, comme lors du basculement d'une version majeure de Debian à la suivante, il faut utiliser **aptitude dist-upgrade** (mise à jour de la distribution). Cela effectue la mise à jour même s'il y a des paquets obsolètes à supprimer et de nouvelles dépendances à installer. C'est également la commande employée par ceux qui exploitent quotidiennement la version *Unstable* de Debian et suivent ses évolutions au jour le jour. Elle est si simple qu'elle parle d'elle-même : c'est bien cette fonctionnalité qui a fait la renommée d'APT.

aptitude dist-upgrade est un synonyme de **aptitude full-upgrade** (mais **apt-get** ne connait que la première variante).

Options de configuration

Outre les éléments de configuration déjà mentionnés, il est possible de configurer quelques aspects d'APT en ajoutant des directives dans un fichier du répertoire `/etc/apt/apt.conf.d/`. Rappelons par exemple qu'il est possible pour APT d'indiquer à **dpkg** d'ignorer les erreurs de collision de fichiers en précisant `DPkg::Options { "--force-overwrite"; }`.

Si l'accès au Web n'est possible qu'à travers un mandataire (proxy), il faut ajouter une ligne semblable à `Acquire::http::proxy "http://`*`monproxy`*`:3128"`. Pour un proxy FTP, on écrira `Acquire::ftp::proxy "ftp://`*`monproxy`*`"`. Découvrez par vous-même les autres options de configuration en consultant la page de manuel `apt.conf(5)`, avec la commande **man apt.conf** (les pages de manuel seront détaillées au chapitre suivant).

Gérer les priorités associées aux paquets

Une des problématiques les plus importantes dans la configuration d'APT est la gestion des priorités des différentes sources de paquets. Il arrive en effet assez fréquemment qu'on souhaite compléter une distribution d'un

B.A.-BA Répertoire en .d

Les répertoires de suffixe `.d` sont de plus en plus souvent employés. Chacun abrite des fichiers ventilant un fichier de configuration. Ainsi, tous les fichiers contenus dans `/etc/apt/apt.conf.d/` constituent les instructions de configuration d'APT. APT les inclura dans l'ordre alphabétique, de sorte que les derniers pourront modifier un élément de configuration défini dans l'un des premiers.

Cette structure apporte une certaine souplesse à l'administrateur de la machine et aux mainteneurs de paquets. En effet, l'administrateur peut facilement modifier la configuration du logiciel en déposant un fichier tout prêt dans le répertoire en question sans devoir modifier de fichier existant. Les mainteneurs de paquets ont la même problématique lorsqu'ils doivent adapter la configuration d'un autre logiciel pour assurer une parfaite cohabitation avec le leur. Mais la charte Debian interdit explicitement toute modification de fichiers de configuration relevant d'autres paquets, interdiction justifiée par le fait que seuls les utilisateurs sont habilités à intervenir ainsi. Rappelons en effet que **dpkg** invite l'utilisateur, lors d'une installation, à choisir la version du fichier de configuration qu'il souhaite conserver lorsqu'une modification y est détectée. Toute modification externe du fichier déclencherait une telle requête, qui ne manquerait pas de perturber l'administrateur certain de n'avoir rien altéré.

En l'absence de répertoire `.d`, il est impossible à un paquet externe d'adapter les réglages d'un logiciel sans en modifier le fichier de configuration. Il doit alors inviter l'utilisateur à intervenir lui-même, en documentant les opérations à effectuer dans le fichier `/usr/share/doc/`*`paquet`*`/README.Debian`.

Selon les applications, le répertoire `.d` est directement exploité, ou géré par un script externe qui en concaténera tous les fichiers pour créer le fichier de configuration à proprement parler. Il est alors important d'exécuter ce script après toute intervention dans ce répertoire pour que les plus récentes modifications soient prises en compte. De même, on prendra soin de ne pas travailler directement sur le fichier de configuration construit automatiquement, sous peine de tout perdre lors de l'exécution suivante du script. Le choix de la méthode (répertoire `.d` utilisé directement ou fichier généré à partir de ce répertoire) est généralement dicté par des contraintes de mise en œuvre, mais dans les deux cas les gains en termes de souplesse de configuration compensent largement les petites complications. Comme exemple de la méthode du fichier généré, on peut citer le serveur de messagerie Exim 4, dont la configuration peut être découpée en plusieurs fichiers (`/etc/exim4/conf.d/*`) qui sont agrégés en un seul (`/var/lib/exim4/config.autogenerated`) par la commande **`update-exim4.conf`**.

ou deux paquets plus récents issus de *Testing*, *Unstable*, ou *Experimental*. Il est possible d'affecter une priorité à chaque paquet disponible (un même paquet pouvant recevoir plusieurs priorités, selon sa version ou sa distribution d'appartenance). Ces priorités dicteront à APT son comportement : pour chaque paquet, il sélectionnera systématiquement la version de plus haute priorité (sauf si cette version est plus ancienne que celle installée et que la priorité associée est inférieure à 1000).

APT définit un certain nombre de priorités par défaut. Chaque version de paquetage déjà installée a une priorité de 100, une version non installée reçoit une priorité de 500 sauf si elle fait partie de la distribution cible (*Target Release*), qu'on spécifie avec l'option `-t` ou la directive `APT::Target-Release`, auquel cas sa priorité passe à 990.

On modifiera ces priorités en intervenant sur le fichier `/etc/apt/preferences` pour y ajouter des entrées de quelques lignes décrivant le nom du ou des paquets concernés, leur version, leur origine, et leur nouvelle priorité.

APT refusera toujours d'installer une version antérieure d'un paquet (portant un numéro de version inférieur à celui de la version actuelle), sauf si la priorité du paquet concerné est supérieure à 1000. APT installera toujours la version de priorité la plus élevée. Si deux versions ont la même priorité, APT installe la plus récente (de numéro de version le plus grand). Si deux paquets de même version ont la même priorité mais diffèrent par leurs contenus, APT installe la version qui n'est pas installée (cette règle doit couvrir le cas d'une mise à jour de paquet sans incrément — normalement indispensable — du numéro de révision).

Concrètement, un paquet de priorité inférieure à 0 ne sera jamais installé. Un paquet de priorité comprise entre 0 et 100 ne sera installé que si aucune autre version du même paquet n'est installée. Avec une priorité comprise entre 100 et 500, le paquet ne sera installé que s'il n'en existe aucune version plus récente, installée ou disponible dans une autre distribution. Un paquet de priorité entre 500 et 990 ne sera installé qu'à défaut de version plus récente, installée ou disponible dans la distribution cible. Une priorité entre 990 et 1000 fera installer le paquet, sauf si la version installée est plus récente. Une priorité supérieure à 1000 provoquera l'installation du paquet, même si cela force APT à installer une version plus ancienne que la version actuelle.

CAS PARTICULIER Priorité d'Experimental

Si vous avez inscrit *Experimental* dans votre fichier `sources.list`, les paquets correspondants ne seront quasiment jamais installés, leur priorité APT étant de 1. C'est un cas particulier qui évite que les gens installent des paquets *Experimental* par erreur, et les oblige à opérer en tapant **`aptitude install`** ***`paquet`*****`/experimental`** — ils ont donc pleinement conscience des risques encourus. Il est possible, mais ce n'est *pas* recommandé, de considérer les paquets *Experimental* comme ceux des autres distributions en leur affectant une priorité de 500 grâce à une entrée dans le fichier `/etc/apt/preferences` :

```
Package: *
Pin: release a=experimental
Pin-Priority: 500
```

Quand APT consulte le fichier `/etc/apt/preferences`, il prend d'abord en compte les entrées les plus précises (souvent, celles spécifiant le paquet concerné) puis les plus génériques (incluant par exemple tous les paquets d'une distribution). Si plusieurs entrées génériques existent, la première correspondant au paquet dont on cherche la priorité est utilisée. Les critères de sélection disponibles comprennent notamment le nom du paquet et la source d'où il provient. Chaque source de paquets est identifiée par un ensemble d'informations contenues dans un fichier `Release`, qu'APT télécharge en même temps que les fichiers `Packages.gz`. Ce dernier spécifie l'origine (habituellement « Debian » pour les paquets des miroirs officiels, mais il peut s'agir du nom d'une personne ou d'un organisme proposant une archive de paquets Debian) ; il précise également le nom de la distribution (habituellement *Stable*, *Testing*, *Unstable* ou *Experimental* pour les distributions standards fournies par Debian) ainsi que sa version (par exemple 5.0 pour Debian *Lenny*). Étudions-en la syntaxe précise au travers de quelques cas vraisemblables d'emploi de ce mécanisme.

Supposons qu'on souhaite utiliser exclusivement des paquets provenant de la version stable de Debian, sans jamais installer ceux des autres versions sauf demande explicite. Il est possible d'écrire ce qui suit dans le fichier `/etc/apt/preferences`:

```
Package: *
Pin: release a=stable
Pin-Priority: 900

Package: *
Pin: release o=Debian
Pin-Priority: -10
```

`a=stable` précise le nom de la distribution concernée. `o=Debian` restreint l'entrée aux paquets dont l'origine est « Debian ». Le terme *pin* (épingle en anglais), est généralement traduit, dans ce contexte, par « étiquetage », car il permet d'accrocher à un paquet une étiquette désignant de quelle distribution il doit provenir.

Supposons maintenant que nous disposions d'un serveur ayant installé de nombreux programmes spécifiques à la version 5.8 de Perl et que l'on veuille s'assurer qu'aucune mise à jour n'en installera une autre version. On peut pour cela utiliser cette entrée :

```
Package: perl
Pin: version 5.8*
Pin-Priority: 1001
```

La documentation de référence sur ce fichier de configuration est disponible dans la page de manuel `apt_preferences(5)`, accessible par la commande **`man apt_preferences`**.

ASTUCE

Commentaires dans /etc/apt/preferences

Il n'existe pas de syntaxe standard pour introduire des commentaires dans le fichier `/etc/apt/preferences`, mais il est possible d'y expliquer le rôle de chaque entrée à l'aide d'un ou plusieurs champs « `Explanation` » (explication) placés en début de bloc :

```
Explanation: Le paquet
  ➥ xserver-xorg-video-i810
  ➥ contenu dans
Explanation: experimental peut être
  ➥ utilisé
Package: xserver-xorg-video-i810
Pin: release a=experimental
Pin-Priority: 500
```

Travailler avec plusieurs distributions

L'outil formidable qu'est **`aptitude`** incite fortement à mettre en place des paquets provenant d'autres distributions. Ainsi, après avoir installé une version *Stable*, vous voulez tester un logiciel présent dans *Testing* ou *Unstable*, sans trop vous éloigner de son état initial.

Même si vous n'êtes pas complètement à l'abri de bogues d'interactions entre les paquets de différentes distributions, **`aptitude`** se révèle fort heureusement très habile pour gérer une telle cohabitation et en minimiser les risques. La meilleure manière de procéder est de préciser toutes les distributions employées dans le fichier `/etc/apt/sources.list` (certains y placent toujours les trois distributions, mais rappelons que l'utilisation d'*Unstable* est réservée aux utilisateurs expérimentés) et de préciser votre distribution de référence avec le paramètre `APT::Default-Release` (voir section « Mise à jour » page 95).

Supposons que *Stable* soit votre distribution de référence, mais que *Testing* et *Unstable* apparaissent également dans votre fichier `sources.list`. Dans ce cas, vous pouvez employer **`aptitude install`** ***`paquet`*****`/testing`** pour installer un paquet depuis *Testing*. Si l'installation échoue parce que certaines dépendances ne peuvent pas être satisfaites, autorisez-le à satisfaire ces dernières dans *Testing* en ajoutant le paramètre `-t testing`. Il en ira évidemment de même pour *Unstable*.

Dans cette situation, les mises à jour (« **`safe-upgrade`** » et « **`dist-upgrade`** ») ont lieu dans le cadre de *Stable* sauf pour les paquets mis à jours depuis une autre distribution : ces derniers suivront les dernières évolutions dans celles-là. Nous donnons ci-dessous l'explication de ce comportement grâce aux priorités automatiques employées par **APT**. N'hésitez pas à employer **`apt-cache policy`** (voir encadré) pour vérifier les priorités indiquées.

ASTUCE apt-cache policy

Pour mieux comprendre le mécanisme des priorités, n'hésitez pas à employer **`apt-cache policy`** pour voir la priorité par défaut associée à chaque source de paquets, et **`apt-cache policy`** ***`paquet`*** pour consulter les priorités des différentes versions disponibles d'un paquet donné.

Tout est lié au fait que **APT** ne considère que les paquets de version supérieure ou égale à la version installée (sauf configuration particulière dans `/etc/apt/preferences` forçant la priorité de certains paquets au-delà de 1000).

Considérons un premier paquet installé depuis *Stable* et qui en est à la version 1, dont la version 2 se trouve dans *Testing* et la 3 dans *Unstable*. La version installée a une priorité de 100, mais la version disponible dans *Stable* (la même) a une priorité de 990 (en tant que version dans la distribution cible). Les paquets de *Testing* et *Unstable* ont une priorité de 500 (priorité par défaut d'une version non installée). Le vainqueur est donc la version 1 avec une priorité de 990. Le paquet « reste dans *Stable* ».

Prenons le cas d'un autre paquet, dont la version 2 a été installée depuis *Testing* ; la version 1 est disponible dans *Stable* et la 3 dans *Unstable*. La version 1 (de priorité 990 — donc inférieure à 1000) est ignorée car plus petite que la version installée. Restent donc les versions 2 et 3, toutes deux de priorité 500. Face à ce choix, **APT** choisit la version plus récente, celle de la distribution *Unstable*. Si vous ne souhaitez pas qu'un paquet installé depuis *Testing* puisse migrer vers *Unstable* il faut associer une priorité inférieure à 500 (par exemple, 490) aux paquets provenant d'*Unstable* en modifiant `/etc/apt/preferences` :

```
Package: *
Pin: release a=unstable
Pin-Priority: 490
```

Commande apt-cache

La commande **apt-cache** permet de consulter un certain nombre d'informations stockées dans la base de données interne d'APT. Ces informations — qui constituent une sorte de *cache* — sont rassemblées depuis les différentes sources données dans le fichier `sources.list` au cours de l'opération **aptitude update**.

> **VOCABULAIRE Cache**
>
> Un cache (« antémémoire », en français officiel) est un système de stockage temporaire servant à accélérer des accès fréquents à des données lorsque la méthode d'accès normale est coûteuse (en termes de performances). Cette notion s'applique dans de très nombreuses situations et à différentes échelles, depuis le cœur des microprocesseurs jusqu'aux systèmes de stockage de grande capacité.
> Dans le cas d'APT, les fichiers `Packages` de référence sont ceux situés sur les miroirs Debian. Cependant, il serait très inefficace de devoir passer à travers le réseau pour chaque recherche que l'on souhaite faire dans la base de données des paquets disponibles. APT stocke donc (dans `/var/lib/apt/lists/`) une copie de ces fichiers, et les recherches se font à l'aide de ces fichiers locaux. De même, `/var/cache/apt/archives/` contient un cache des paquets déjà téléchargés, ce qui évite de les télécharger de nouveau si on souhaite les réinstaller après les avoir supprimés.

Le programme apt-cache permet notamment de rechercher des paquets à l'aide de mots-clés, en tapant **apt-cache search *mot-clé***. On peut aussi consulter les en-têtes des différentes versions disponibles d'un paquet avec **apt-cache show *paquet***. Cette commande produira la description du paquet ainsi que ses dépendances, le nom de son mainteneur, etc. Signalons que **aptitude search** et **aptitude show** fonctionnent de manière similaire.

Certaines fonctions ne servent que bien plus rarement. Ainsi, **apt-cache policy** permet de consulter les priorités des différentes sources de paquets ainsi que celles des paquets qui bénéficient d'un traitement particulier. On peut encore citer **apt-cache dumpavail** qui affiche les en-têtes de toutes les versions disponibles de tous les paquets. **apt-cache pkgnames** affiche une liste de tous les paquets existants dans la mémoire *cache*.

Frontaux : aptitude, synaptic, gnome-apt

APT est un programme C++ dont la majorité du code est déportée dans la bibliothèque partagée **libapt-pkg**. La raison de ce choix est qu'il rend relativement facile de réaliser une interface (un « frontal »), puisqu'il suffit de faire appel au code placé dans la bibliothèque. **apt-get** n'était d'ailleurs à l'origine qu'un frontal développé pour tester **libapt-pkg**, bien que son succès ait tendance à le faire oublier.

aptitude

aptitude est un programme interactif en mode semi-graphique, utilisable sur la console, qui permet de naviguer dans la liste des paquets installés et disponibles, de consulter l'ensemble des informations, et de les marquer en vue d'une installation ou d'une suppression. Comme il s'agit cette fois d'un programme réellement conçu pour être utilisé par les administrateurs, on y trouve des comportements par défaut plus intelligents que dans **apt-get**, en plus d'une interface plus abordable.

```
 Actions  Annuler  Paquet  Solutions  Rechercher  Options  Vues  Aide
C-T : Menu  ? : Aide  q : Quitter  u : M-à-J  g : Téléch./Inst./Suppr. Pqts
aptitude 0.4.3
--\ Paquets installés
  --\ admin - Utilitaires d'administration (installation de logiciels, gestion d
    --\ main - L'archive principale de Debian
i     adduser                                               3.99        3.99
i     apt                                                   0.6.46.2    0.6.46.2
i     apt-utils                                             0.6.46.2    0.6.46.2
i     aptitude                                              0.4.3-1     0.4.3-1
i     base-files                                            4           4
i     base-passwd                                           3.5.11      3.5.11
i     cron                                                  3.0pl1-99   3.0pl1-99
i     debconf                                               1.5.8       1.5.8
i     debconf-i18n                                          1.5.8       1.5.8
i A   deborphan                                             1.7.23      1.7.23

Ces paquets sont actuellement installés sur votre ordinateur.
```

Figure 6–1
Gestionnaire de paquets aptitude

En ce qui concerne l'interface, **aptitude** présente initialement une vue séparant les paquets selon leur état actuel (installé, non installé, ou installé mais non disponible sur les miroirs — d'autres sections affichent les tâches, les paquets virtuels, et les paquets apparus récemment sur les miroirs). Afin de faciliter la navigation thématique, il est possible d'employer d'autres vues. Dans tous les cas, **aptitude** affiche sur un écran une liste combinant catégories et paquets. Les catégories étant organisées dans une arborescence, on pourra déplier ou refermer les branches respectivement avec les touches *Entrée*, *[* et *]*. On utilisera + pour marquer un paquet comme à installer, - pour le marquer comme à supprimer, et _ pour le purger (à noter que ces touches peuvent aussi être utilisées sur les catégories, auquel cas les actions concernées seront appliquées à tous les paquets de la catégorie) ; *u* met à jour les listes de paquets disponibles, et *Shift + u* prépare une mise à jour globale du système. *g* bascule vers un résumé des modifications demandées (et un nouvel appui sur *g* déclenche alors la mise en application), et *q* permet de sortir de la vue courante ; si l'on est dans la vue initiale, cela équivaut à fermer **aptitude**.

Pour chercher un paquet, on utilisera /, suivi d'un motif de recherche. Ce motif peut porter sur le nom du paquet, mais aussi sur sa description (si on le fait précéder de `~d`), sa section (avec `~s`) ou d'autres caractéristiques détaillées dans la documentation. Les mêmes motifs peuvent servir à filtrer les paquets affichés, fonctionnalité accessible grâce à la touche *l*.

DOCUMENTATION aptitude

Nous n'entrons pas ici dans tous les détails de l'utilisation de ce frontal, en nous contentant de donner le minimum de survie. **aptitude** est relativement bien documenté, et l'on consultera donc le mode d'emploi complet, qui est disponible lorsque le paquet `aptitude-doc-fr` est installé.

- file:///usr/share/doc/aptitude/html/fr/index.html

Suivi des paquets installés automatiquement

Une des fonctionnalités majeures d'**`aptitude`** (qui a été reprise dans la version d'**`apt-get`** de *Lenny*) est le suivi des paquets qui n'ont été installés que par le jeu des dépendances. Ces paquets sont dits « automatiques » et marqués par un A dans la liste des paquets ; on y trouvera souvent des bibliothèques, par exemple. Lorsque l'on supprime un paquet du système, les paquets automatiques correspondants sont également sélectionnés pour la suppression. Il est possible de forcer le caractère automatique d'un paquet (par *Shift +m*) ou de le désactiver (touche *m*). Une saine habitude, lorsque l'on maintient un système avec **`aptitude`**, consiste à marquer comme automatiques tous les paquets dont on n'a pas un besoin direct, afin qu'ils soient automatiquement supprimés lorsqu'ils ne sont plus nécessaires. On pourra pour cela soit naviguer dans la liste des paquets installés et jouer du *Shift + m*, soit appliquer cet attribut sur des sections entières (par exemple la section `libs`). Cette habitude, en plus d'aider à garder un système propre, présente l'avantage de permettre de visualiser simplement les paquets en usage sur une machine, sans que cette liste soit polluée par toutes les bibliothèques et dépendances ; le motif consacré, à utiliser avec *l* (pour passer en mode filtrage), est `~i!~M`, qui spécifie que l'on ne souhaite afficher que les paquets actuellement installés (`~i`) mais non marqués comme automatiques (`!~M`).

Il arrive que l'on veuille savoir pourquoi un paquet automatiquement installé est présent sur le système. Pour obtenir cette information directement depuis la ligne de commande, on peut employer **`aptitude why`** ***`paquet`*** :

```
$ aptitude why python-notify
i gnome                    Dépend system-config-printer (>= 1.0.0)
i A system-config-printer Dépend python-notify
```

OUTIL aptitude en ligne de commande

La plupart des fonctionnalités d'**`aptitude`** sont accessibles aussi bien par l'interface interactive que par la ligne de commande, et cette ligne de commande ne dépaysera pas trop les habitués d'**`apt-get`** et **`apt-cache`**.
Mais les fonctionnalités évoluées d'**`aptitude`** se retrouvent également sur la ligne de commande. On retrouve ainsi les mêmes motifs de recherche de paquets qu'en version interactive. Ainsi, si on veut faire dans les paquets le ménage automatique mentionné ci-contre, et qu'on sait qu'aucun programme localement installé n'a besoin de bibliothèques particulières ou de modules Perl, on pourra marquer les paquets correspondants comme automatiques en une seule commande :

```
# aptitude markauto '~slibs|~sperl'
```

On voit ici la puissance du système de motifs de recherche d'**`aptitude`**, qui permet de sélectionner d'un coup l'ensemble des paquets des sections `libs` et `perl`.
Attention, s'il existe des paquets que cette commande marque comme automatiques, et qu'aucun autre paquet n'en dépend, ils seront immédiatement supprimés (avec une demande de confirmation).

À SUIVRE Évolutions récentes de apt-get et aptitude

Certains des avantages que présentait historiquement **`aptitude`** par rapport à **`apt-get`** ont été gommés. En particulier, dans la version *Lenny* de Debian, **`apt-get`** mémorise les paquets qui ne sont installés que pour satisfaire les dépendances, de la même manière qu'**`aptitude`** l'a toujours fait. Il est également capable de suivre les recommandations exprimées par un paquet sur un autre. Parmi les évolutions récentes d'**`aptitude`**, citons également l'apparition d'une version avec une interface graphique. Cette fonctionnalité, encore en cours de développement, n'a pas été suffisamment stabilisée pour être intégrée dans *Lenny* et n'est, à l'heure où nous écrivons ceci, disponible que dans *experimental*. Gageons qu'elle migrera vers *unstable* suffisamment tôt pour être stabilisée et faire partie de Squeeze.

> ALTERNATIVE **deborphan et debfoster**
>
> Avant l'apparition d'**`aptitude`** et de son suivi des paquets automatiques, il existait deux utilitaires qui permettaient de déterminer une liste de paquets non nécessaires, **deborphan** et **`debfoster`**.
> **deborphan**, le plus rudimentaire des deux, recherche simplement dans les sections `libs` et `oldlibs` (à défaut d'instructions supplémentaires) les paquets actuellement installés dont aucun autre paquet installé ne dépend. Cette liste peut ensuite servir de point de départ pour supprimer les paquets inutiles.
> **`debfoster`** a une approche plus évoluée, qui se rapproche un peu de celle d'**`aptitude`** : il maintient une liste de paquets installés explicitement, et se rappelle d'une invocation sur l'autre quels paquets sont réellement requis. Si de nouveaux paquets sont apparus sur le système, et que **`debfoster`** ne les connaît pas comme des paquets requis, ils seront présentés à l'écran, ainsi qu'une liste de leurs dépendances. Le programme propose alors un choix, permettant de supprimer le paquet (ainsi que ceux dont il dépend, le cas échéant), de le marquer comme explicitement requis, ou de l'ignorer temporairement.

Gestion des recommandations, suggestions et tâches

Un autre intérêt d'**`aptitude`** est qu'il respecte les recommandations entre paquets tout en permettant à l'utilisateur de ne pas les installer au cas par cas. Ainsi, le paquet `gnome-desktop-environment` recommande (entre autres) `gnome-accessibility`. Si l'on sélectionne le premier pour l'installation, le second sera également sélectionné (et marqué comme automatique s'il n'est pas déjà présent sur le système). Un appui sur *g* permet de s'en rendre compte : `gnome-accessibility` figure sur l'écran de résumé des actions en attente dans la liste des paquets ajoutés automatiquement pour satisfaire des dépendances. On peut cependant décider de ne pas l'installer, en le désélectionnant avant de valider les opérations.

On notera que cette fonction de suivi des recommandations ne s'applique pas lors d'une mise à jour. Ainsi, si une nouvelle version de `gnome-desktop-environment` recommande un paquet qu'il ne recommandait pas auparavant, il ne sera pas marqué pour l'installation. En revanche il sera mentionné dans l'écran de mise à jour, afin de vous laisser la possibilité de l'installer malgré tout.

Les suggestions entre paquets sont également prises en compte, mais de manière adaptée à leur statut particulier. Ainsi, comme `gnome-desktop-environment` suggère `gnome-audio`, ce dernier sera listé sur l'écran de résumé des actions (dans la section des paquets qui sont suggérés par d'autres paquets), afin qu'il soit visible et que l'administrateur puisse décider de tenir compte, ou non, de la suggestion. Mais comme il s'agit d'une simple suggestion et non d'une dépendance ou recommandation, le paquet ne sera pas sélectionné, et sa sélection devra être manuelle (et le paquet ne sera donc pas marqué comme automatique).

Dans la même veine, rappelons qu'`aptitude` exploite intelligemment le concept de tâche. Ces tâches étant affichées comme des catégories dans les écrans de listes de paquets, on peut soit choisir une tâche complète à installer ou supprimer, soit consulter la liste des paquets inclus dans une tâche afin d'en sélectionner un sous-ensemble plus limité.

Meilleurs algorithmes de résolution

Enfin, signalons pour terminer cette section qu'**`aptitude`** dispose d'algorithmes plus évolués qu'**`apt-get`** en ce qui concerne la résolution des situations délicates. Si un ensemble d'actions est demandé, mais que ces actions, menées conjointement, aboutissent à un système incohérent, **`aptitude`** évalue plusieurs scénarios possibles et les propose par ordre de pertinence décroissante. Ces algorithmes ne sont cependant pas infaillibles ; heureusement, il reste la possibilité de sélectionner manuellement les actions à effectuer. Si les actions actuellement sélectionnées mènent à des contradictions, le haut de l'écran mentionne un nombre de paquets « cassés » (et on peut naviguer directement vers ces paquets en appuyant sur *b*). Il est alors possible de construire manuellement une solution aux problèmes constatés. On peut notamment, en sélectionnant un paquet avec *Entrée*, avoir accès aux différentes versions disponibles. Si le choix d'une de ces versions plutôt que d'une autre permet de résoudre le problème, on n'hésitera pas à utiliser cette fonction. Lorsque le nombre de paquets cassés descendra à zéro, on pourra en toute confiance passer par le résumé des actions à effectuer pour une dernière vérification avant leur mise en application.

NOTE **Journal d'aptitude**

De même que **dpkg**, **`aptitude`** garde dans son journal (`/var/log/aptitude`) la trace des actions effectuées. Cependant, comme les deux commandes fonctionnent à un niveau bien différent, on ne trouve pas les mêmes informations dans les journaux respectifs. Là où celui de **dpkg** liste pas à pas les opérations exécutées sur chaque paquet individuel, celui d'**`aptitude`** donne une vue d'ensemble sur les opérations de plus haut niveau comme une mise à jour globale du système. Attention, ce journal ne contient que le résumé des opérations initiées par **`aptitude`**. Si l'on utilise occasionnellement d'autres frontaux (voire directement **dpkg**), le journal d'**`aptitude`** n'aura qu'une vision partielle des choses, et on ne pourra pas s'en servir pour reconstituer un historique fiable du système.

synaptic

`synaptic` est un gestionnaire de paquets Debian en mode graphique (il utilise GTK+/GNOME).

Il dispose d'une interface graphique efficace et propre. Ses nombreux filtres prêts à l'emploi permettent de voir rapidement les nouveaux paquets disponibles, les paquets installés, ceux que l'on peut mettre à jour, les paquets obsolètes, etc. En naviguant ainsi dans les différentes listes, on indique progressivement les opérations à effectuer (installer, mettre à jour, supprimer, purger). Un simple clic suffit à valider l'ensemble de ces choix, et toutes les opérations enregistrées sont alors effectuées en une seule passe.

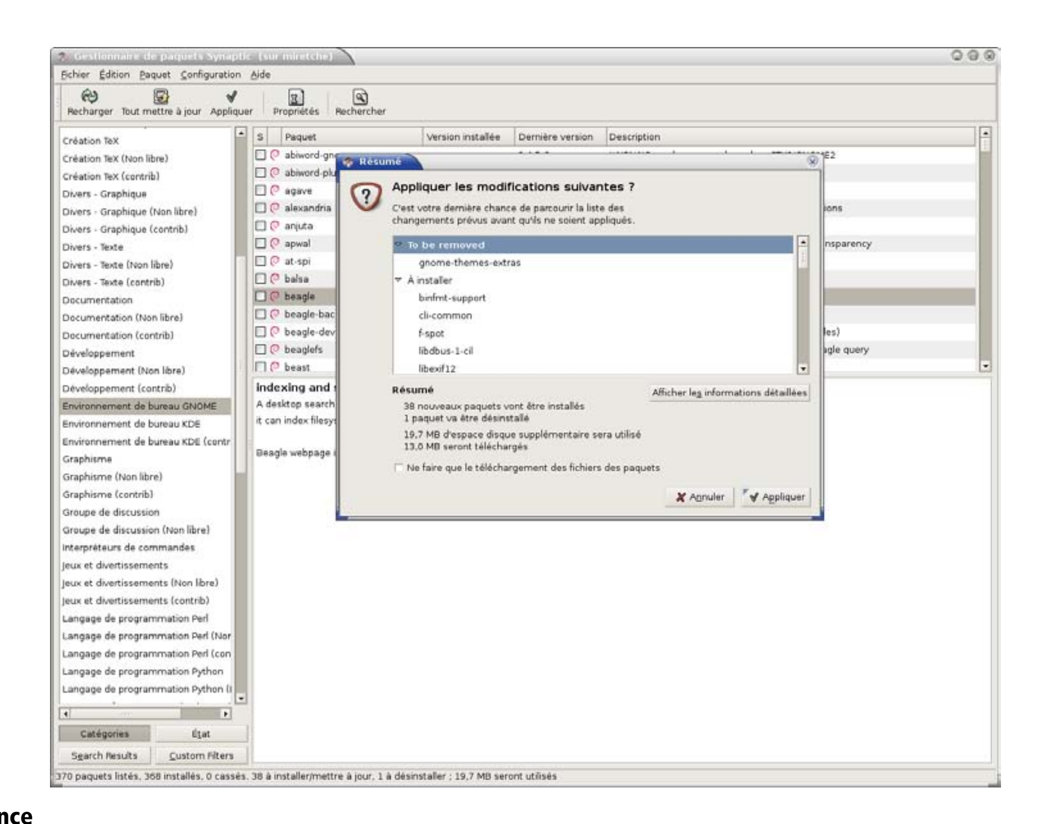


Figure 6–2
Gestionnaire de paquets synaptic

EN PRATIQUE
Ajouter des clés de confiance

Lorsqu'une source de paquets tierce est ajoutée au fichier `sources.list` il faut désormais porter à la connaissance de APT la clé de confiance correspondante (ou sinon il se plaindra constamment qu'il ne peut pas vérifier l'authenticité des paquets contenus dans le dépôt concerné). Pour cela, il faut avant tout récupérer la clé en question : la plupart du temps elle sera fournie sous la forme d'un fichier texte (qui sera nommé `cle.asc` dans les exemples ci-dessous).

On peut alors ajouter cette clé de confiance par la commande **`apt-key add <cle.asc`** (à exécuter avec les droits administrateur). Une autre manière de procéder existe avec l'interface graphique **`synaptic`** : l'onglet *Authentication* dans le menu *Configuration>Dépôts* permet d'importer le fichier `cle.asc` préalablement récupéré.

Pour ceux qui préfèrent une application dédiée et veulent plus de détails sur les clés de confiance, il est possible d'employer le programme **`gui-apt-key`** (du paquet éponyme). Il s'agit d'une petite interface graphique qui gère le trousseau de clés de confiance.

Vérification d'authenticité des paquets

Étant donné l'importance qu'accordent les administrateurs de Falcot SA à la sécurité, ils veulent s'assurer de n'installer que des paquets garantis provenant de Debian et non altérés en cours de route. En effet, un pirate pourrait tenter d'agir indirectement sur des machines en modifiant un paquet Debian diffusé afin d'y ajouter les instructions de son choix. Lorsque le paquet ainsi modifié sera installé, ces instructions agiront, par exemple afin de dérober les mots de passe. C'est pourquoi Debian offre un moyen de s'assurer que le paquet installé provient bien de son mainteneur et qu'il n'a subi aucune modification par un tiers : il existe un mécanisme de scellement des paquets.

Cette signature n'est pas directe : le fichier signé est un fichier `Release` placé sur les miroirs Debian et qui donne la liste des différents fichiers `Packages` (y compris sous leurs formes compressées `Packages.gz` et `Packages.bz2`, et les versions incrémentales), accompagnés de leur somme de contrôle MD5 (pour vérifier que leur contenu n'a pas été altéré). Ces fichiers `Packages` renferment à leur tour une liste de paquets

Debian et leurs sommes de contrôle MD5, afin de garantir que leur contenu n'a pas lui non plus été altéré.

La gestion des clés de confiance se fait grâce au programme **apt-key**, fourni par le paquet apt. Ce programme maintient à jour un trousseau de clés publiques GnuPG, qui sont utilisées pour vérifier les signatures des fichiers Release.gpg obtenus depuis les miroirs Debian. Il est possible de l'utiliser pour ajouter manuellement des clés supplémentaires (si l'on souhaite ajouter des miroirs autres que les miroirs officiels) ; mais dans le cas le plus courant, on n'a besoin que des clés officielles Debian, qui sont automatiquement maintenues à jour par le paquet debian-archive-keyring (qui appelle **apt-key** lors de son installation et de sa mise à jour). Cependant, la première installation de ce paquet est également sujette à caution, car même s'il est signé comme les autres paquets, cette signature ne peut pas être vérifiée extérieurement. On s'attachera donc à vérifier les empreintes (*fingerprints*) des clés importées, avant de leur faire confiance pour installer de nouveaux paquets :

```
# apt-key fingerprint
/etc/apt/trusted.gpg
pub   1024D/6070D3A1 2006-11-20 [expire: 2009-07-01]
    Empreinte de la clé = A999 51DA F9BB 569B DB50  AD90 A70D AF53 6070 D3A1
uid                  Debian Archive Automatic Signing Key (4.0/etch) <ftpmaster@debian.org>
pub   1024D/ADB11277 2006-09-17
    Empreinte de la clé = 7EA3 91D7 2477 203B 58C0  4FBC B5D0 C804 ADB1 1277
uid                  Etch Stable Release Key <debian-release@lists.debian.org>
pub   1024D/BBE55AB3 2007-03-31 [expire: 2010-03-30]
    Empreinte de la clé = 6039 406A 4EDC E124 CF08  7B0A EC61 E0B0 BBE5 5AB3
uid                  Debian-Volatile Archive Automatic Signing Key (4.0/etch)
sub   2048g/36CA98F3 2007-03-31 [expire: 2010-03-30]
pub   1024D/F42584E6 2008-04-06 [expire: 2012-05-15]
    Empreinte de la clé = 7F5A 4445 4C72 4A65 CBCD  4FB1 4D27 0D06 F425 84E6
uid                  Lenny Stable Release Key <debian-release@lists.debian.org>
```

Une fois ces clés placées dans le trousseau, APT effectuera systématiquement les vérifications des signatures avant toute opération risquée ; les frontaux sont alors en mesure d'afficher un avertissement si l'on demande à installer un paquet dont l'authenticité n'a pu être vérifiée.

Mise à jour d'une distribution à la suivante

Un des éléments les plus marquants de Debian est sa capacité à mettre à jour un système d'une distribution stable vers la suivante (le fameux *dist-upgrade*, qui a contribué à la réputation du projet). Avec un peu d'attention, on peut ainsi migrer un ordinateur en quelques minutes ou dizaines de minutes, selon la rapidité d'accès aux sources de paquets.

B.A.-BA Notes de publication

Les notes de publication (*release notes*) d'un logiciel ou d'un système d'exploitation sont un document, généralement court par rapport à la documentation complète, qui permet de se faire une idée du logiciel en question, et particulièrement sur la version concernée. Ces documents donnent souvent un résumé des nouvelles fonctionnalités offertes par rapport aux versions précédentes, des instructions de mise à jour, des avertissements pour les utilisateurs des anciennes versions, et parfois des errata.
Pour les versions de Debian, on trouvera ces notes de publication sur le Web, autant pour la version stable courante que pour les précédentes (qui restent accessibles par leur nom de code) :

- http://www.debian.org/releases/stable/releasenotes
- http://www.debian.org/releases/etch/releasenotes

Démarche à suivre

Comme le système Debian a le temps d'évoluer entre deux versions stables, on prendra soin de lire, avant d'entreprendre la mise à jour, les notes de publication.

Nous allons ici nous attacher particulièrement à la migration d'un système *Etch* en *Lenny*. Comme toute opération majeure sur un système, cette mise à jour comporte une certaine part de risque, et il est donc vivement conseillé de s'assurer que les données importantes sont sauvegardées avant de s'engager dans la procédure.

Pour faciliter (et raccourcir) la mise à jour, il est également recommandé de faire un peu de nettoyage dans les paquets installés, pour ne garder que ceux qui sont réellement nécessaires. Pour cela, on mettra à profit les fonctions d'**aptitude**, éventuellement en conjonction avec **deborphan** et **debfoster** (voir page 104). On pourra par exemple utiliser la commande suivante :

```
# deborphan | xargs aptitude remove
```

Passons à la mise à jour du système. On commencera par indiquer à APT qu'il doit utiliser *Lenny* au lieu de *Etch*, en modifiant le fichier `/etc/apt/sources.list` en conséquence. Si ce fichier ne contient que des références à *Stable* et non à un de ces noms de code, c'est encore plus simple : la modification n'est pas nécessaire, puisque *Stable* est toujours identique à la dernière version publiée de Debian. Dans les deux cas, on n'oubliera pas de rafraîchir la base de données des paquets disponibles (**aptitude update**, ou le bouton de mise à jour dans **synaptic**).

Une fois que ces nouvelles sources de paquets sont référencées, on mettra à jour les paquets `aptitude` et `apt` ; les versions fournies dans *Etch* présentent en effet des limitations qui risqueraient d'interrompre la mise à jour du système.

Pour éviter de se retrouver avec un système qui ne démarre plus, on prendra ensuite soin de mettre à jour (ou d'installer) les paquets les plus cruciaux :

- le chargeur de démarrage `grub` (parfois `lilo`) ;
- les outils permettant de construire le « disque virtuel d'initialisation » ou *initrd* : `initramfs-tools` ;
- la bibliothèque standard : `libc6` ou une de ses variantes optimisées pour certains processeurs comme `libc6-i686` ;
- le système de gestion des fichiers de périphériques : `udev` ;
- enfin, le noyau : on installera, selon le matériel dont on dispose, `linux-image-486`, `linux-image-686` ou `linux-image-686-bigmem`. Ces paquets correspondent uniquement à l'architecture i386, ceux qui disposent

d'une machine basée sur une autre architecture devront sélectionner d'autres noyaux (citons notamment `linux-image-2.6-amd64` pour AMD64, et les `linux-image-powerpc*` pour les machines PowerPC).

Une fois ces préliminaires accomplis, on pourra passer à la mise à jour proprement dite, que ce soit avec **aptitude** ou **synaptic**. On vérifiera les actions à effectuer avant de les déclencher (pour éventuellement ajouter des paquets suggérés, ou désélectionner des paquets qui ne sont que recommandés) ; le frontal devrait dans tous les cas arriver à un scénario dont la situation finale est un système *Lenny* cohérent et à jour. Il suffira alors de patienter durant le téléchargement des paquets, de répondre aux questions Debconf, et de regarder la magie s'opérer pendant le reste de la procédure en gardant un œil attentif sur les éventuelles questions portant sur le remplacement de fichiers de configuration qui auraient été localement modifiés.

Gérer les problèmes consécutifs à une mise à jour

Malgré tous les efforts des mainteneurs Debian, une mise à jour majeure du système d'exploitation cause parfois quelques soucis. Les nouvelles versions de certains logiciels sont parfois incompatibles avec les précédentes (évolution d'un format de données, comportement par défaut qui diffère, etc.). En outre, certains bogues passent inaperçus malgré la période de test précédant la publication d'une nouvelle version.

Pour anticiper les problèmes liés aux évolutions des logiciels mis à jour, il est utile d'installer le paquet `apt-listchanges`. Il affichera, au début d'une mise à jour de paquet, des informations relatives aux embarras possibles. Ces informations sont rédigées par les mainteneurs de paquet à l'intention des utilisateurs et placées dans des fichiers `/usr/share/doc/`*`paquet`*`/NEWS.Debian`, et en tenir compte évitera toute mauvaise surprise.

Parfois, la nouvelle version d'un logiciel ne fonctionne plus du tout. C'est par exemple le cas si le logiciel n'est pas très populaire et n'a pas été suffisamment testé ; une mise à jour de dernière minute peut aussi introduire des régressions qui ne sont découvertes qu'après publication. Dans ce cas, le premier réflexe sain est de consulter le système de suivi de bogue à l'adresse `http://bugs.debian.org/`*`paquet`* pour déterminer si le problème est déjà connu et signalé. Si ce n'est pas le cas, il faut le signaler avec **reportbug**. Sinon, la lecture du rapport de bogue sera généralement très instructive :

- on peut y découvrir l'existence d'un correctif qui permet alors de recompiler une version corrigée du paquet Debian (voir page 368) ;
- parfois d'autres utilisateurs ont trouvé un moyen de contourner le problème et partagent leur expérience dans l'historique du bogue ;
- enfin un paquet corrigé peut avoir été préparé par le mainteneur et être disponible en téléchargement.

Selon la gravité du bogue, une nouvelle version peut être préparée pour être intégrée dans une nouvelle révision de la version stable. Dans ce cas, un paquet corrigé est peut-être disponible dans la section `proposed-updates` des miroirs Debian. On peut alors temporairement ajouter

```
deb http://ftp.fr.debian.org/debian lenny-proposed-updates main contrib non-free
```

dans son fichier `sources.list` et installer la mise à jour avec **`apt-get`** ou **`aptitude`**. Le nom canonique pour *Stable* est actuellement `proposed-updates`, mais l'emploi de `lenny-proposed-updates` est plus cohérent car `etch-proposed-updates` existe aussi.

Si le paquet n'est pas encore disponible dans cette section, on peut vérifier s'il est en attente de validation par les SRM (les gestionnaires de la version stable) en consultant leur page web. Les paquets listés sur cette page ne sont pas encore disponibles publiquement mais l'on sait au moins que le processus de publication suit son cours.

- http://release.debian.org/proposed-updates/stable.html

Maintenir un système à jour

Debian est une distribution qui évolue au fil du temps. Bien que les changements soient surtout visibles dans les versions *Testing* et *Unstable*, même la version *Stable* voit quelques modifications de temps en temps (il s'agit principalement de correctifs pour des problèmes de sécurité). Quelle que soit la version installée, il est souvent utile de rester à jour, pour profiter des dernières évolutions et des corrections de bogues.

Bien sûr, il est possible de lancer régulièrement un outil permettant de vérifier l'existence de paquets mis à jour, puis de déclencher l'opération. Cependant, c'est une tâche fastidieuse et répétitive, surtout si l'on a plusieurs machines à administrer. Il existe heureusement des outils permettant d'automatiser une partie des opérations.

Citons tout d'abord **`apticron`**, dans le paquet du même nom. Il s'agit simplement d'un script, appelé quotidiennement par **`cron`**, qui met à jour la liste des paquets disponibles et envoie un courrier électronique à une adresse donnée pour lister les paquets qui ne sont pas installés dans leur dernière version, ainsi qu'une description des changements qui ont eu lieu. Ce script vise principalement les utilisateurs de Debian *Stable*, on s'en doute : ces mails seraient quotidiens et vraisemblablement très longs sur les versions plus mobiles de Debian. Lorsque des mises à jour sont disponibles, **`apticron`** les télécharge, mais ne les installe pas. L'administrateur peut ainsi exécuter la mise à jour plus rapidement, puisque les paquets sont déjà dans le cache d'APT, il ne sera plus nécessaire d'attendre qu'ils transitent depuis la source de paquets.

Si l'on administre plusieurs machines, il est certes intéressant d'être prévenu lorsque certaines ont besoin d'une mise à jour, mais cette opération elle-même peut rester fastidieuse. On pourra donc tirer parti du script /etc/cron.daily/apt, installé par le paquet apt. Ce script est lui aussi lancé quotidiennement par **cron**, donc sans interface interactive. Pour contrôler son fonctionnement, on utilisera des variables de configuration d'APT (qui seront donc stockées dans un fichier sous /etc/apt/apt.conf.d/). Les trois plus importantes sont :

APT::Periodic::Update-Package-Lists

Cette option permet de spécifier une fréquence (en jours) de mise à jour des listes de paquets. Si l'on utilise **apticron**, on pourra s'en passer, puisque cela ferait double emploi.

APT::Periodic::Download-Upgradeable-Packages

Cette option spécifie également une fréquence en jours, qui porte sur le téléchargement des paquets mis à jour. Là encore, les utilisateurs d'**apticron** pourront s'en passer.

APT::Periodic::AutocleanInterval

Enfin, cette option couvre une fonction que n'a pas **apticron** : elle spécifie la fréquence à laquelle le cache d'APT pourra être automatiquement épuré des paquets obsolètes (ceux qui ne sont plus disponibles sur les miroirs ni référencés par aucune distribution). Elle permet de ne pas avoir à se soucier de la taille du cache d'APT, qui sera ainsi régulée automatiquement.

D'autres options permettent de jouer plus finement sur le comportement du nettoyage de cache ; nous ne les aborderons pas ici, mais elles sont décrites dans le script /etc/cron.daily/apt lui-même.

Ces outils conviennent très bien pour des serveurs, mais pour un ordinateur de bureau, on préférera en général un mécanisme plus interactif. C'est pourquoi la tâche « Environnement graphique de bureau » référence **update-notifier** et **update-manager**. Le premier est une petite application qui affiche une icône dans la zone de notification d'un environnement de bureau lorsque des mises à jour sont disponibles. Dans ce cas, un clic sur cette icône lance **update-manager**, une interface simplifiée pour effectuer des mises à jour. Elle permet de naviguer dans les mises à jour disponibles, de lire le changelog et la description des paquets concernés, et de décider individuellement si une mise à jour doit être installée ou non. Notons au passage que le paquet fournit des éléments de configuration afin que /etc/cron.daily/apt mette à jour la liste des paquets et télécharge les mises à jour disponibles. Loin de la puissance d'**aptitude** et de **synaptic**, ce tandem ne gère que les mises à jour des paquets déjà installés, et offre par son interface minimaliste peu de risques de rendre le système incohérent.

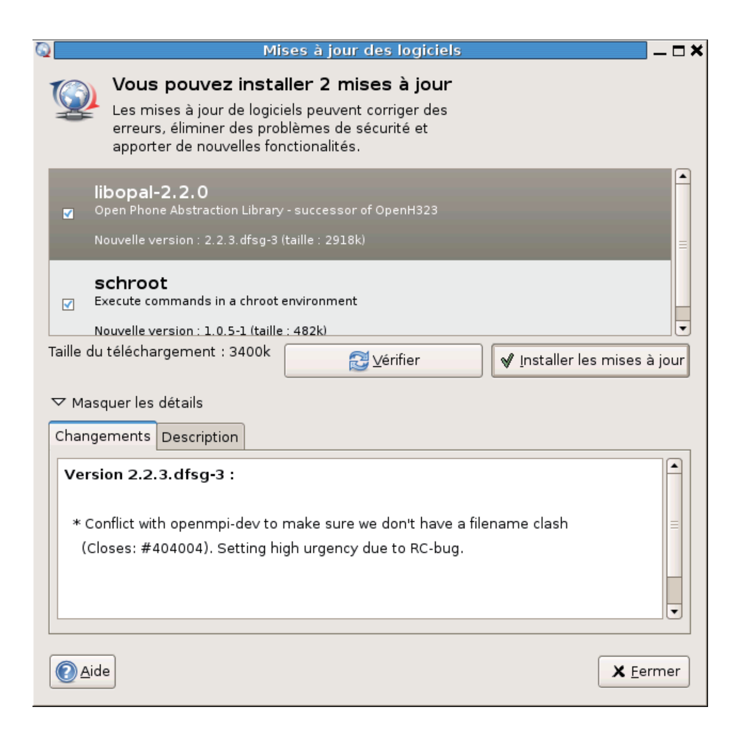


Figure 6–3
Mise à jour avec update-manager

Mise à jour automatique

Dans le contexte de Falcot SA, qui inclut de nombreuses machines et des ressources humaines limitées, les administrateurs souhaitent automatiser au maximum les mises à jour. Les programmes chargés de ces opérations doivent donc fonctionner sans intervention humaine.

Configuration de dpkg

Nous avons déjà vu (en page 76) comment interdire à **dpkg** de demander confirmation du remplacement d'un fichier de configuration (avec les options `--force-confdef --force-confold`). Il reste trois éléments à prendre en compte : les interactions générées par APT lui-même, celles provenant de **debconf**, et les interactions en ligne de commande intégrées dans les scripts de configuration des paquets.

Configuration d'APT

En ce qui concerne APT, la réponse est simple. Il suffit de lui préciser l'option `-y` ou `--assume-yes`, qui répondra « oui » automatiquement à toutes les questions qu'il aurait pu poser.

Configuration de debconf

Pour **debconf**, la réponse mérite un plus long développement. Dès sa naissance, ce programme fut prévu pour permettre de vérifier la pertinence et le volume des questions posées à l'utilisateur, ainsi que la manière dont elles le seront. C'est pourquoi sa configuration demande la priorité minimale à partir de laquelle **debconf** posera une question. Quand il s'interdit d'interroger l'humain, ce programme utilise automatiquement la valeur par défaut définie par le mainteneur du paquet. Il faut encore choisir une interface pour l'affichage des questions (frontal, ou *frontend* en anglais).

Parmi la liste des interfaces possibles, `noninteractive` (non interactive) est très particulière : la choisir désactive toute interaction avec l'utilisateur. Si un paquet désire malgré tout lui communiquer une note d'information, celle-ci sera automatiquement transformée en courrier électronique.

Pour reconfigurer **debconf**, on utilise l'outil **dpkg-reconfigure** inclus dans le paquet **debconf** ; la commande est **dpkg-reconfigure debconf**. Il est aussi possible de changer temporairement les choix de configuration effectués à l'aide de variables d'environnement (DEBIAN_FRONTEND permet ainsi de changer d'interface, comme expliqué dans la page de manuel `debconf(7)`).

Gestion des interactions en ligne de commande

Finalement, les interactions en ligne de commande des scripts de configuration exécutés par **dpkg** sont les plus difficiles à éliminer. Il n'existe en effet aucune solution standard, et aucune réponse n'est meilleure qu'une autre.

La solution généralement employée est de supprimer l'entrée standard (en y redirigeant le contenu de `/dev/null`, par exemple avec la syntaxe ***commande* </dev/null**), ou d'y brancher un flux continu de retours à la ligne. Cette méthode n'est pas fiable à 100 % mais elle permet en général d'accepter les choix par défaut, puisque la plupart des scripts interprètent l'absence de réponse explicite comme une validation de la valeur proposée par défaut.

La combinaison miracle

Si l'on met bout à bout les éléments de configuration exposés dans les sections précédentes, il est possible de rédiger un petit script capable d'effectuer une mise à jour automatique assez fiable.

EXEMPLE Script pour mise à jour non interactive

```
export DEBIAN_FRONTEND=noninteractive
yes '' | apt-get -y -o Dpkg::Options::="--force-confdef" -o
  ➥ Dpkg::Options::="--force-confold" dist-upgrade
```

EN PRATIQUE Le cas de Falcot SA

Les administrateurs de Falcot doivent faire avec un système informatique hétérogène, dont les machines servent des buts différents. Ils choisiront donc pour chaque machine la solution la plus adaptée.
En pratique, les serveurs (sous *Lenny*) utiliseront la « combinaison miracle » évoquée ci-dessus, pour être maintenus à jour automatiquement. Seuls les plus critiques (les pare-feu, par exemple) seront configurés pour utiliser **apticron**, afin que les mises à jour ne se fassent que sous le contrôle d'un administrateur.
Les postes bureautiques du service administratif (en *Lenny* aussi) seront configurés avec le tandem **update-notifier**/**update-manager**, de sorte que les utilisateurs pourront déclencher les mises à jour eux-mêmes ; il est en effet important qu'elles se fassent à l'initiative des utilisateurs principaux des ordinateurs, sans quoi des modifications imprévues et silencieuses (donc mystérieuses) pourraient les perturber.
Enfin, pour les quelques ordinateurs du laboratoire qui utilisent *Testing* pour bénéficier des dernières versions des logiciels, les administrateurs de Falcot décident simplement de configurer APT pour qu'il prépare périodiquement les mises à jour, sans les effectuer. De cette manière, lorsqu'ils voudront mettre à niveau (manuellement) ces machines expérimentales, ils pourront se concentrer sur les actions réellement utiles, les phases fastidieuses de téléchargement ayant déjà été effectuées automatiquement.

Recherche de paquets

Avec la quantité énorme, et sans cesse croissante, de logiciels distribués par Debian, il se manifeste un paradoxe : lorsque l'on a un besoin, la quantité de paquets disponibles rend parfois difficile la recherche d'un paquet correspondant à ce besoin. Il existe, mais il est enfoui si profond sous une myriade d'autres qu'il est introuvable. Le besoin d'outils de recherche de paquets s'est donc fait de plus en plus criant au fil du temps. Il semble que ce problème est en passe d'être résolu.

La recherche la plus triviale correspond à une recherche sur le nom exact d'un paquet. Si **apt-cache show *paquet*** renvoie un résultat, c'est que le paquet existe. Malheureusement, il n'est pas toujours facile de deviner le nom du paquet.

On peut aussi effectuer des recherches textuelles sur les noms des paquets, mais cela ne fait pas beaucoup avancer les choses. On n'atteint quelque chose de réellement utilisable qu'avec les recherches sur les descriptions : chaque paquet ayant, en plus de son nom, une description plus ou moins détaillée, une recherche par mots-clés pourra souvent rap-

ASTUCE Conventions de nommage de certains paquets

Certaines catégories de paquets suivent une nomenclature conventionnelle qui peut permettre de deviner le nom du paquet Debian. Par exemple, pour les modules Perl, la convention dicte qu'un module publié en amont sous le nom `XML::Handler::Composer` sera empaqueté en tant que `libxml-handler-composer-perl`. Les modules ajoutant au noyau 2.6.26-1 le système de fichiers *squashfs*, compilés pour un processeur de type 686, sont dans le paquet `squashfs-modules-2.6.26-1-686`. Il n'est hélas pas possible d'établir une convention de nommage pour tous les paquets, même si le responsable essaie généralement de rester au plus près du nom choisi par le développeur amont.

porter des résultats. On utilisera pour cela **apt-cache** ; par exemple, **apt-cache search video** renverra la liste de tous les paquets contenant le mot-clé « video » dans leur nom ou leur description.

Si l'on souhaite effectuer des recherches plus complexes, on pourra utiliser **aptitude**, qui permet de spécifier une expression logique portant sur différents champs des paquets. Par exemple, on pourra obtenir la liste des paquets dont le nom contient `kino`, la description `video` et le nom du responsable `paul` :

```
$ aptitude search kino~dvideo~mpaul
p   kino  - Non-linear editor for Digital Video data
$ aptitude show kino
Paquet : kino
État : non installé
Version : 1.3.0-2+lenny1
Priorité : supplémentaire
Section : graphics
Responsable : Paul Brossier <piem@debian.org>
Taille décompressée : 9638k
Dépend: libasound2 (> 1.0.16), libatk1.0-0 (>= 1.20.0),
        libavc1394-0 (>= 0.5.3), libavcodec51 (>= 0.svn20080206-8) |
        libavcodec-unstripped-51 (>= 0.svn20080206-8), libavformat52
        […]
Recommande: ffmpeg
Suggère: udev | hotplug, vorbis-tools, sox, mjpegtools, lame
Est en conflit: kino-dvtitler, kino-timfx, kinoplus
Remplace: kino-dvtitler, kino-timfx, kinoplus
Fournit: kino-dvtitler, kino-timfx, kinoplus
Description : Non-linear editor for Digital Video data
  Kino allows you to record, create, edit, and play movies recorded with DV
  camcorders. This program uses many keyboard commands for fast navigating
  and editing inside the movie.
  The kino-timfx, kino-dvtitler and kinoplus sets of plugins, formerly
  distributed as separate packages, are now provided with Kino.
  Site : http://www.kinodv.org/

Étiquettes: hardware::camera, implemented-in::c, implemented-in::c++,
    interface::x11, role::program, scope::application,
    suite::gnome, uitoolkit::gtk, use::editing,
    works-with::video, x11::application
```

Le résultat ne contient ici qu'un paquet, `kino`, qui satisfait bien les trois conditions requises.

Ces recherches multi-critères manquent un peu de flexibilité, et ne sont donc pas toujours utilisées au maximum de leur puissance. Il a donc été mis en place un système de « marqueurs » ou « étiquettes » (en anglais, *tags*), qui propose une autre approche de la recherche. Ces étiquettes correspondent à une classification thématique des paquets selon plusieurs axes, appelée « classification à facettes ». Pour reprendre l'exemple de `kino` ci-dessus, on constate que ce paquet se présente sous la forme

d'une interface graphique (qui utilise GTK), qu'il s'agit d'un logiciel Gnome, que sa fonction principale est l'édition, et qu'il travaille sur des données de type vidéo.

Il est alors possible de naviguer dans cette classification, à la recherche d'un paquet correspondant aux besoins, ou du moins d'un petit nombre de paquets parmi lesquels on pourra faire le tri manuellement. Pour cela, on pourra soit utiliser le motif de recherche `~G` dans **aptitude**, soit plus simplement naviguer vers le site qui centralise les étiquettes : http://debtags.alioth.debian.org/cloud/ Si l'on sélectionne les étiquettes `works-with::video` et `use::editing`, on obtient une poignée de paquets, dont les éditeurs vidéo `kino` et `pitivi`. Ce système de classement est appelé à se généraliser et les gestionnaires de paquets intégreront progressivement des interfaces pour rechercher efficacement les logiciels.

En résumé, selon la complexité des recherches que l'on souhaite mener, on utilisera un programme adapté :

- **apt-cache** ne permet que les recherches textuelles dans le nom et la description des paquets, il est très pratique pour retrouver rapidement le nom précis d'un paquet qu'on peut facilement décrire avec quelques mots clés bien ciblés ;
- pour des recherches portant également sur les relations entre paquets et le nom du responsable, on pourra utiliser **synaptic** ;
- si l'on souhaite ajouter une recherche par étiquettes, on se dirigera vers **packagesearch** ; interface graphique dont le seul but est de mener des recherches dans la liste des paquets disponibles, selon plusieurs critères ; on peut même chercher des paquets d'après le nom des fichiers qu'ils contiennent ;
- enfin, si l'on a besoin de construire des requêtes complexes avec des opérateurs logiques, on utilisera la très puissante (mais relativement obscure) syntaxe des motifs de recherche d'**aptitude**, aussi bien en ligne de commande qu'en mode interactif.